Lecture Note 5

1 Linear Programming

1.1 General form

We are intereested in solving the following optimization problems: we want to optimize a linear objective function($\sum_j c_j \cdot x_j$, where c_j are constants, x_j are variables) subject to linear inequality/equality constraints. Here is the general form of linear programming.

$$\min \sum_j c_j \cdot x_j$$
$$\text{s.t } \forall i, \sum_j a_{ij} \cdot x_j \geq b_i$$
$$\forall i, x_i \geq 0$$

1.2 Algorithms to solve LP

- Simplex: by Dantzig. This algorithm is fast in practice, but in worst case takes exponential time
- Ellipsoid Algorithm: by Khachiyan in 70's. In polynomial time.
- Interial Point Algorithm: by Karmarkar in early 80's. Also polynomial time.

1.3 Comparing with Integer Programming (IP)

- LP can be solved in polynomial time
- IP is NP-complete

1.4 Example with Traveling Salesman Problem (TSP)

- Target function:

  $$\min \sum_{(i,j) \in E} X_{(i,j)} \cdot d(i,j)$$

  We want $X_{(i,j)}$ to be 1 if and only if $(i,j) \in$ TOUR, and 0 otherwise. We need a set of constraints to make it work
- First try:

  $$\forall v, \sum_i X_{(v,i)} \geq 2, \text{ and } \sum_i X_{(v,i)} \leq 2$$

  May not work, as it might give many disconnected cycles instead of a single tour
- Fix: For every subset, the number of out-degree is at least 2.

2 K-center problem

2.1 Problem description

n points, and want to select a set $S \subseteq$ points where $|S| = k$. Define $\text{cost}(p, S) = \min_{q \in S} d(p, q)$, and we want to minimize

$$\min_S \max_p \text{cost}(p, S)$$

2.2 Gonzalez Algorithm

- $S_1 \leftarrow$ any point
- For $i = 2$ to k do
  - $S_i \leftarrow$ point with highest cost relative to $\{S_1, \ldots, S_{i-1}\}$

2.3 Streaming

link

The stream would be $x_1, x_2, x_3, \ldots$. The subset we are maintaining is $C = \{c_1, c_2, c_3, \ldots, c_{k'}\}$, where $k' \leq k$. r_p is a lower bound on the optimal solution.

2.4 Algorithm

- $C =$ First k distinct points
- $r_0 = 0$, $p = 1$
- For stage p (x_i arrives)
  - If $d(x_i, C) \leq 4r_{p-1}$, then forget x_i
  - Else
    - Add x_i to C
    - If $|C| > k$
      - Let $r_p = \min_{c_j, c_l \in C} \frac{d(c_j, c_l)}{2}$
      - <u>Recluster</u> (C, r_p)
      - $p = p + 1$, and we move to the next stage

2.4.1 <u>Recluster</u> (C, r_p)

- $C' \leftarrow C$
- For all pair $c_j, c_l \in C'$
  - If $d(c_j, c_l) \leq 4 \cdot r_p$
    - Drop c_j or c_l from C'.
- // Let C' is now a <u>maximal subset</u> of C such that $\forall c_j, c_l \in C'$ have dist $> 4 \cdot r_p$.
- $C \leftarrow C'$

2.4.2 Final answer:

Subset C, and distance $8r_p$.

2.5 Analysis

We have the following properties

- **Lemma 1**: $\forall c_j, c_l \in C, d(c_j, c_l) \geq 4r_{p-1}$.
- **Lemma 2**: $r_p \geq 2r_{p-1}$
- **Lemma 3**: $\max_x d(x, C) \leq 8r_p$

Lemma 1 guarantees that nodes in C are not too far from each other, **Lemma 2** guarantees that dropped points are not too far from C. **Lemma 3** guarangees that r_p increases quickly.

The correctness of first lemma is trivial, our Recluster procedure guarrantees this. Now we prove **Lemma 2** with **Lemma 1**. We prove it by induction on p. When $p = 1$, it is obvious that $r_1 \geq r_0 = 0$. Suppose this lemma is true when $p = t$, we prove it is also true when $p = t + 1$.

Since it is true when $p = t$, we have $r_t \geq 2r_{t-1}$. We moved from stage t to stage $t + 1$ because we have to merge at least one pair of node in C. So $r_{t+1} = \min_{c_j, c_l \in C} \frac{d(c_j, c_l)}{2} \geq \frac{4 \cdot r_t}{2} = 2r_t$, and we are done.

As for **Lemma 3**, this property will hold as long as we are not reclustering. At stage p, we have $\max_x d(x, C) \leq 8 \cdot r_p$, which means the farthest node x from C is closer than $8r_p$ (or $d(x, c_j) \leq 8r_p$ for some $c_j \in C$). When we move to the next stage $p + 1$, c_j might have been dropped, since its distance to some other c_l is less than $4r_{p+1}$. So the distance from x to c_l is upper bounded by $8r_p + 4r_{p+1} \leq 4r_{p+1} + 4r_{p+1} = 8r_{p+1}$.